TapGazer: Text Entry with Finger Tapping and Gaze-directed Word Selection

Zhenyi He
New York University
New York, United States
zhenyi.he@nyu.edu

Christof Lutteroth
University of Bath
Bath, United Kingdom
c.lutteroth@bath.ac.uk

Ken Perlin
New York University
New York, United States
perlin@cs.nyu.edu

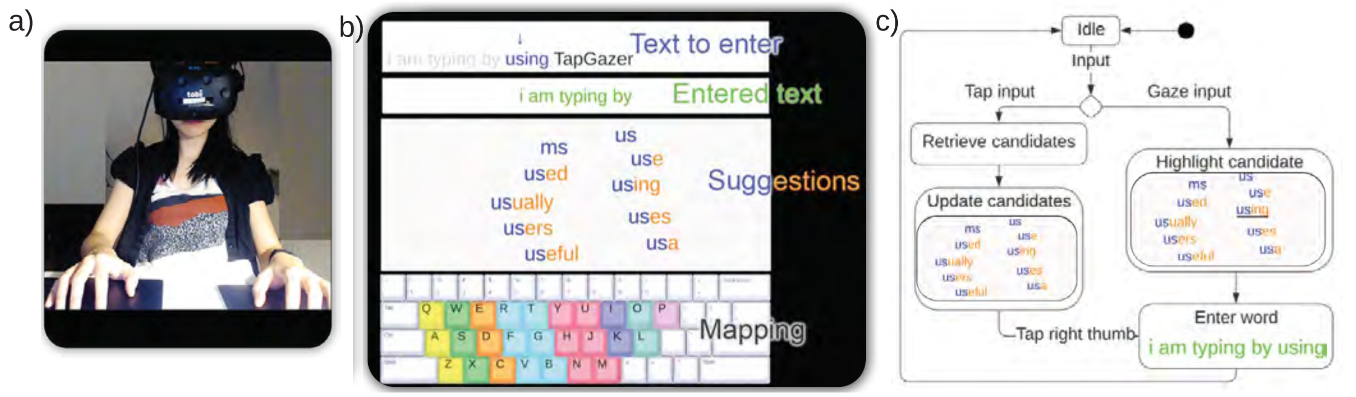


Figure 1: a) Physical setup: A VR user enters text without needing to see hands or keyboard, by tapping on a surface and resolving ambiguity between candidate words via gaze selection. b) Visual interface: Fingers are mapped to multiple letters (see colors at bottom); the central area shows candidate words corresponding to the current input sequence of finger taps. Users can select a word by gazing at it and tapping the right thumb. c) State machine of Tapgazer with gaze selection and word completion.

ABSTRACT

While using VR, efficient text entry is a challenge: users cannot easily locate standard physical keyboards, and keys are often out of reach, e.g. when standing. We present TapGazer, a text entry system where users type by tapping their fingers in place. Users can tap anywhere as long as the identity of each tapping finger can be detected with sensors. Ambiguity between different possible input words is resolved by selecting target words with gaze. If gaze tracking is unavailable, ambiguity is resolved by selecting target words with additional taps. We evaluated TapGazer for seated and standing VR: seated novice users using touchpads as tap surfaces reached 44.81 words per minute (WPM), 79.17% of their QWERTY typing speed. Standing novice users tapped on their thighs with touch-sensitive gloves, reaching 45.26 WPM (71.91%). We analyze TapGazer with a theoretical performance model and discuss its potential for text input in future AR scenarios.

CCS CONCEPTS

• **Human-centered computing** → **Text input**.

KEYWORDS

Input Techniques; Text Entry; Eye Tracking; Virtual Reality; Typing

ACM Reference Format:
Zhenyi He, Christof Lutteroth, and Ken Perlin. 2022. TapGazer: Text Entry with Finger Tapping and Gaze-directed Word Selection. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3491102.3501838

1 INTRODUCTION

Text entry is one of the most frequent, important, and demanding tasks in personal computing. Because efficient text entry methods are crucial to productivity, an enormous amount of research has been conducted on methods that improve their usability. As new types of electronic devices such as smartphones have become available, new text entry methods have been proposed [3, 25, 96]. With the increasing popularity of Virtual Reality (VR), there is an expanding interest in text entry methods that can support VR users [40, 55, 110, 112]. While using VR, efficient text entry poses the following challenges:

Proximity. VR users typically interact with virtual environments using their hands, often turning their bodies to change orientation, and are frequently standing or even walking in their VR usage area. These movements generally take a user's hands away from stationary physical keyboards, and often out of reach of the keys. Therefore, a versatile VR text entry method should allow users to be more mobile than with a standard physical keyboard and also relax the requirement of having to keep fingers aligned with keys. Related works have addressed this by proposing virtual keyboards controlled with VR headsets [103, 104], portable standard keyboards [33, 44, 80], and input methods using hands [50, 101], fingers [26, 36, 52, 67, 74, 94, 108, 109], gaze [48, 49, 65], or stylus [21].

Visibility. VR headsets occlude the real world, making it harder to locate physical keyboards, move to a suitable pose close to the keyboard, and align the fingers with the keys for efficient touch typing. Ideally, a VR text entry method should afford users awareness of physical keyboards, or avoid the use of a physical keyboard in the first place. Related works have addressed this by by providing visual cues about the physical keyboard position [11, 33, 41, 44, 55, 69, 74, 97], attaching keyboards to a user's body so she is kinaesthetically aware of their position [80], or virtual keyboards that readily show up in the user's field of view [19, 53, 106].

Learnability. Because text entry is a basic task of computing systems, it should ideally be easy to learn. Many novel text entry methods require users to learn entirely new, non-standard text entry skills such as new keyboard layouts [7, 63, 85, 113]. However, as many users are already proficient in the use of a QWERTY keyboard, much previous work on VR text entry has aimed to exploit this familiarity to improve learnability [10, 33, 44, 80, 106].

We propose TapGazer, a novel method for casual text entry in VR designed to address these challenges by combining finger tapping and eye gaze input (Figure 1). We envision VR users to use TapGazer if they mainly use their hands to interact naturally in VR (without controllers), but need to enter some text quickly from time to time, e.g. to take notes or send text messages. Users type by tapping their fingers, without needing to look at their hands or be aware of finger position. The location where a finger is tapped is not needed by TapGazer, therefore taps may be detected with any input device capable of discerning which finger is currently being tapped, e.g. finger-worn accelerometers such as TapStrap, touch-sensitive surfaces such as smart cloth, or visual finger tracking like leap motion. This enables users to quickly move from VR hand interaction to text entry without having to align their fingers on keys, and facilitates use of TapGazer on soft surfaces such as thighs and in different poses such as seating or standing. Tracking fingers' identities and detecting whether a finger has tapped is generally less complicated and more accurate than tracking both the identity and location of each finger, and it is generally easier for users to focus on tapping their fingers without the need to worry about finger location. Given a suitable input device, any available surface may be used to support the hands and facilitate tapping movements, e.g. a table or one's thighs.

To enable text entry by finger tapping, TapGazer simplifies keyboard input by assigning multiple letters to each finger. Because this mapping is one-to-many, it is ambiguous (see the color-coded keyboard layout in Figure 1(b)). We resolve this ambiguity by showing word suggestions in the users' display and allowing users to select the correct word via gaze and determine the selection via a thumb tap. TapGazer's finger-to-letter mapping is based on a QWERTY keyboard layout, so people can reuse their QWERTY skills and retain the performance benefits of ten-finger typing, which is generally faster than alternatives such as word-gesture keyboards [17]. TapGazer supports the entry of unknown words, symbols, and cursor navigation by allowing users to switch between different modes. Furthermore, because gaze tracking may not always be available, we describe variants of TapGazer that work without gaze tracking by allowing users to select target words with additional taps. We investigate the following research questions:

- **RQ1** How can text input be efficiently achieved using only finger taps and gaze?
- **RQ2** How does TapGazer perform in terms of speed, accuracy, and user preference?
- **RQ3** How can we model user performance in TapGazer?

We address these questions by first discussing the design of TapGazer (RQ1), then reporting on user studies evaluating TapGazer (RQ2) in seated and standing VR scenarios with different tap sensors, and lastly providing a model-based analysis of how different users of TapGazer will likely perform (RQ3).

Novelty. Some previous work has looked at reduced QWERTY keyboards and word disambiguation. VType [26] applies a reduced keyboard layout, attempting to reconstruct words automatically based on finger sequence, grammar, and context, but does not allow users to choose between ambiguous words. The 1Line keyboard [54] and the stick keyboard [32] flatten the QWERTY keyboard from three rows to one, allowing users to choose between ambiguous words through touchscreen gestures and arrow keys. Yet to the best of our knowledge we are the first to investigate tapping while resolving ambiguity through gaze. The performance we measured for TapGazer (45.26 WPM on average in a standing VR scenario) compares favorably with those reported for similar works (see Table 1). In summary, we make the following key contributions:

(1) A design that combines tap and gaze for effective text entry in VR, with variants for use without gaze tracking and for accommodating different user preferences.
(2) Evidence that TapGazer is usable and easy-to-learn for novice users, and able to reach average speeds of 44.81 WPM (78.81% of their QWERTY typing speed) using touchpads in a sitting VR scenario (n=14) and 45.26 WPM with word completion (71.91%) using touch-sensitive gloves in a standing VR scenario (n=5).
(3) A model-based performance analysis illustrating the effects of different design options and usage strategies.
(4) Open-source software and hardware designs to facilitate future research.

2 RELATED WORK

To develop a fast and usable text entry design using tap and gaze, we closely investigated prior work in alternative keyboard layout design, gaze interaction, and text entry for VR and similar scenarios. An overview of the most relevant and fastest methods, with their average speeds in words per minute (WPM), is shown in Table 1. For works that reported users' QWERTY performance, we list also the percentage of their QWERTY WPM users were able to achieve.

For devices where a full-size physical keyboard is not available, many specialized text entry solutions have been proposed, e.g. for touch screens [43, 54, 88], mobile phones [25, 115], and handheld devices [15]. Moreover, using a finger [9, 76] or pen [45]

Table 1: Summary of prior text entry solutions that are compatible with our usage scenarios, ordered by their average WPMs.

Design	Average WPMs	% of QWERTY keyboard WPM	Examples
Typing QWERTY on a touch surface	17.2–44.6	74.59% [88]	BlindType [58], PalmBoard [107], TOAST [88]
Tapping on tiny surface	11.0–41.0		Ahn & Lee [3], Vertanen et al. [95], VelociTap [96]
Reduced physical QWERTY keyboard	7.3–30.0	37.17% [54]	Stick [32], 1Line [54], LetterWise [61], VType [26]
Gesture typing	15.6–34.2		GestureType [110], Chen et al. [17]s
Mid-air chord gesture typing	22.0–24.7		Sridhar et al. [93], Adhikary [2]
Typing with pinch gestures	11.9–23.4		TipText [105], DigiTouch [101], BiTipText [105]
Mid-air finger tapping	17.8–23.0	49.24% [108]	VISAR [24], ATK [108]
Tapping with head or controller on a soft alternative keyboard	7.25–21.1		PizzaText [112], RingText [104], HiPad [40], Curved QWERTY [106], Boletsis & Kongsvik [10]
Tapping QWERTY with head or controller	11.3–15.6		Tap/Dwell [110]
Gaze typing plus touch	14.6–15.5		EyeSwipe [48, 49], TAGSwipe [47]

for handwritten text input has been considered, although this is slow compared to typing. Speech-to-text is also a widely explored option with the potential to be faster than typing [86]; however, it has limited accuracy and is not always suitable, e.g. when the environment is noisy, other people are talking, or the content is of a sensitive or personal nature.

A key requirement of manual typing approaches is detection and tracking of the fingers. Gloves [52, 94], markers [36, 67], audio signals [98], cameras [84, 109], and specific devices such as Leap Motion [108] have all been investigated. Based on this, various input recognition methods have been proposed, with some recognizing input as single characters ('character-level') and others recognizing entire words ('word-level'). Methods recognizing larger chunks of input (e.g. words, sentences [96]) are typically more effective than those recognizing characters [95]. Input prediction and correction methods can be used to improve the performance of text entry [20, 31, 75, 114].

2.1 Alternative Keyboard Layouts

Some alternative layouts support a limited interaction size with a reduced number of keys, which makes them relevant for TapGazer. A common consideration is the similarity to familiar layouts such as QWERTY or T9 for learnability, e.g. for mobile phones [25, 61], smart glasses [3], and smartwatches [81]. Familiar layouts are often adapted to new typing gestures, e.g. using thumb-to-finger interaction for small-screen devices or VR/AR using split QWERTY [73, 101] or T9 layouts [102]. Another trend is rearranging keyboard characters into different 2D or 3D shapes: QuikWriting [79] and its gaze-version [5] distribute letters into a circle; PizzaText [112], WrisText [30], and HiPad [40] use a pie-shaped layout; Keycube [13] attaches push buttons to a physical magic cube for typing.

When applying a reduced keyboard layout, fingers or keys are not uniquely assigned to characters, so a mechanism for disambiguation becomes necessary. LetterWise [61] uses prefix-based rather than word-based disambiguation, i.e. users press a button if the current character is wrong and then the respective character of the next-likely prefix is shown. By repeatedly pressing the button, even non-dictionary words can be typed. Stick keyboard [32] compresses the QWERTY keyboard into one line, with each key mapped to 2-3 characters. Users choose one of several ambiguous words by scrolling through possible candidates with button presses. Similarly, 1Line keyboard [54] reorganizes the QWERTY keyboard to a single line specifically for touchscreen typing, using touch gestures to support candidate selection. TapGazer is also based on a reduced QWERTY layout, but it uses different mechanisms for faster disambiguation.

2.2 Gaze-assisted Text Entry

Text entry with gaze does not require a physical keyboard; it is a natural option to consider for VR, which can incorporate gaze tracking. Gaze-only methods mainly fall into four categories [66]: direct gaze pointing with dwell ("gaze typing"), eye switches, discrete gaze gestures, and continuous gaze gestures ("gaze writing"). Dwell [6, 38, 65] (i.e. looking at keys for a certain time to trigger clicks) has been widely applied and optimized to solve the Midas Touch problem [39] (i.e. inadvertent clicks). Approaches for reducing the dwell time necessary for each key have been explored, e.g. by dynamically adjusting it based on prefix [64], word frequency, or character placement [70]; however, it is still a major factor slowing down typing speed. Eye-switch approaches try to avoid dwell by using other operations such as blinking, brow interaction or head movements [29] as triggers. Similarly, discrete gaze gestures have been proposed to avoid dwell, e.g. by adding a resting zone in the typing area [5], 'swiping' over a keyboard with gaze to enter a word [16, 48], or using other confirmatory eye movements such as inside-outside-inside saccades [87].Some disambiguation algorithms have been proposed to improve the accuracy of word-level gaze gestures [56, 77]. Dasher [99] uses continuous gaze gestures to zoom towards and select candidate letters and words.

Some approaches try to speed up gaze-only text entry methods by using other modalities for key and word selection, e.g. a brain-computer interface [60], or touch gestures [3, 47]. If gaze tracking is not available, many gaze-based approaches can be modified to use head movement only [104, 110]. This can be combined with other head gestures, e.g. nodding for letter selection [57]. Overall, gaze-based text entry methods facilitate social privacy and can be used while standing or moving in VR [83]; however, they are still much slower than physical keyboards (below 25 WPM) as gaze movements are generally time-consuming [28]. Therefore, TapGazer uses gaze for disambiguation rather than typing.

2.3 Text Entry in VR

Various methods have been investigated for text entry in VR [22]. Because text entry using a physical keyboard is faster than other typing solutions, many approaches for text entry in VR try to facilitate access to a standard physical keyboard rather than replace it. This has mainly been done by tracking and visualizing a physical keyboard in VR while sitting at a desk, either by blending in a video stream showing the real keyboard [11, 41, 55, 69] or by visualizing the keyboard in VR [11, 33, 44, 74, 97]. To support better mobility, HawKEY [80] uses a portable keyboard for users to type on while standing and walking in VR. These approaches show that using a physical keyboard and high-quality tracking can lead to good performance. However, using a physical keyboard can be cumbersome and break immersion when interacting naturally with a virtual environment through body movements, e.g. when standing.

In order to integrate text entry more closely with natural VR interaction, pointing gestures on virtual keyboards have been investigated. Xu et al. [103] and Speicher et al. [92] compared pointing methods to selecting virtual keys with controllers, head, and hand. Boletsis & Kongsvik [10] proposed virtual keyboard layouts to optimize controller-based key selection. PizzaText [112] arranges virtual keys in a circle separated into segments. Didehkhorshid et al. [21] compared controller-based with stylus-based virtual keyboard interaction. Yanagihara et al. [106] introduced a curved virtual QWERTY keyboard, allowing users to use a controller to swipe between different keys. Similarly, Chen et al. [17] proposed word gestures by pointing and swiping at a virtual keyboard. Additionally, Dube & Arif [23] researched the impact of key design on virtual keyboards for typing speed and accuracy. While these approaches improve mobility, similar to what TapGazer aims to do, they are much slower than physical keyboards, typically below 25 WPM.

Some VR text entry methods use fingers or hands directly. A popular approach is to detect pinch gestures between fingers and thumbs, e.g. using a data glove. Pinch keyboard [12] combines pinch with hand rotation and position to select letters. KITTY [46] uses pinch gestures on different parts of the thumb. PinchType uses a reduced keyboard, and if necessary, allows the user to disambiguate words with hand gestures [27]. DigiTouch [101] uses continuous touch position and pressure. Quadmetric [50] and HiFinger [42] support one-handed text entry with pinch. RotoSwype [35] uses one-handed word gestures by rotating a ring worn on one hand. Yu et al. propose one-dimensional 'handwriting' of words with a tracked finger or controller [111]. Such pinch and word gesture based approaches are flexible but slow, with typical speeds far below 20 WPM. Also, mid-air finger gestures can be hard to track and can lead to fatigue when performing longer tasks [2, 24].

Some approaches for eyes-free typing could be feasible for use in VR scenarios although they were not originally designed for VR. BlindType [58] allows users to type without looking at the typing interface using single-thumb touchpad gestures. PalmBoard [107] provides a one-handed touch typing solution that decodes which keys users likely intend to type on a flat touchpad. Similarly, TOAST [88] leverages statistical decoding algorithms for ten-finger typing on flat touch-sensitive surfaces.

Some approaches use finger touch or taps similar to TapGazer. FaceTouch [34] allows users to type on a touch surface attached to their headset. ARKB [51] proposes visual tracking of fingers for tapping on a virtual QWERTY keyboard. VISAR [24] facilitates mid-air one-finger tapping on an AR QWERTY keyboard. VType [26] uses finger tapping on a reduced QWERTY keyboard layout and reconstructs words based on finger sequence, grammar, and context for text input in VR. The accuracy reported for a predefined vocabulary is high; however, no method for disambiguation between candidate words was considered and no typing speed was reported. VType, the 1Line keyboard [54] and the stick keyboard [32], which all involve tapping on a reduced QWERTY keyboard, are the works closest to TapGazer. Tapping on a reduced QWERTY keyboard is promising for text entry in VR as it is flexible and robust compared to alternatives. Therefore, we explore how it can be optimized by using gaze input and additional taps for disambiguation.

3 TAPGAZER DESIGN

TapGazer allows users to tap words as if they are typing them on a physical QWERTY keyboard and then to disambiguate their tap input by selecting their target word through gaze. It was designed primarily for VR users, but could also be useful for other scenarios where more conventional input devices are unavailable or difficult to access. Given suitable sensors, users can type by tapping their fingers on any surface or even in mid-air. As TapGazer only considers the identity of the finger that is currently tapping and not its position, it only needs to know which of the user's 10 fingers has just been tapped, if any, at any given time. Each of the 26 letters of the alphabet is mapped to at least one of the eight non-thumb fingers, while the two thumbs are reserved for controlling editing functions for word selection, undoing a selection, deletion and cursor navigation. Figure 1 illustrates the state machine of TapGazer with gaze selection and word completion. Starting from an idle state, TapGazer waits for tap or gaze input events. Except for the thumbs, a finger tap adds a letter to the *input string*, starting from an empty string. The input string is constructed from an *input alphabet* with one character for each of the eight fingers: we are using the characters `asdfjkl;`, which correspond to the rest positions of each finger on a QWERTY keyboard, for later reference. When typing a word with TapGazer, the user taps the fingers as they would do when typing on a QWERTY keyboard. However, as each finger tap can be interpreted as one of several characters, the word represented by the input is ambiguous: for example, `fjd` is the input string for the words 'the' and 'bye'. We refer to a set of words that all have the same tapping input string as a *homograph set*. A tap with the left thumb deletes the current input string so users can start the word again. A tap of the right thumb selects the word to enter from a list of suggestions while the word is pointed at by the user's gaze.

As a user enters an input string, the central area of TapGazer's user interface shows a list of word *candidates*: similar to predictive text on a mobile phone, the user is given a list of the most likely words to choose from. TapGazer shows all words in the *homograph set* for the given input string, which we call *complete candidates* as

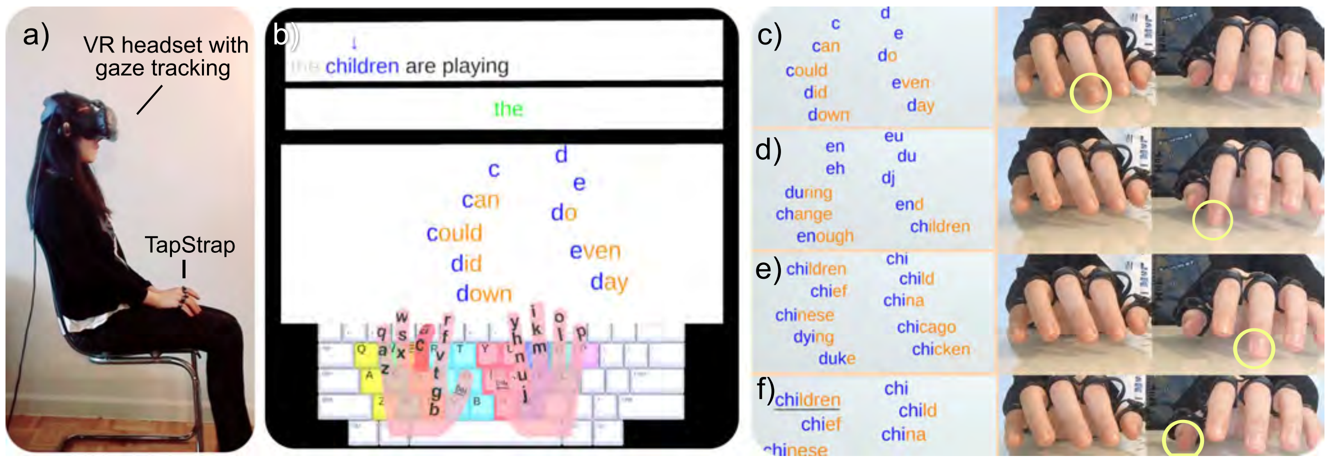


Figure 2: TapGazer text entry example: a) A user is 'typing' on her thighs using a TapStrap device instead of a keyboard. b) The user just started to 'type' the word "children". The interface provides optional visualizations of the finger-key mapping as a virtual keyboard and/or hands. c) The user first tapped the left middle finger (mapped to 'c'), then d) the right index finger (mapped to 'h'), and e) the right middle finger (mapped to 'i'). f) Finally, the user looks at the word "children", which gets highlighted with an underline as it is gazed upon, and taps the right thumb to select the highlighted word.

they are based on the whole input string (e.g. 'the' for `fjd`). Additionally, TapGazer uses a language model to show the most likely *incomplete candidates*, i.e. words with a prefix matching the current input string (e.g. 'these' for `fjd`). After each tap, TapGazer updates the candidates shown. In order to select a candidate, the user looks at it, and in response, the fixated candidate is highlighted with an underline. If the right thumb is tapped, the currently highlighted candidate is selected and added to the entered text. At this point, the TapGazer state machine starts again with an empty input string. If the user taps the right thumb but does not fixate any candidate, then the most likely candidate is selected based on a language model. Figure 2 illustrates how to type 'children' with TapGazer. Word completion in TapGazer can be disabled; in this variant, only complete candidates are shown if they exist. If no complete candidate exists, we show the shortest incomplete candidate to inform users about the progress of typing. Furthermore, we have designed a purely manual variant of TapGazer without gaze tracking, allowing users to disambiguate candidates with extra taps. Figure 4 illustrates different input devices (left) and variants (as decision nodes in the state machine) of Tapgazer.

Several design decisions were made: First, we use finger tapping so that users can 'type' on any surface and require no context knowledge between the surface location and finger/hand location. Second, we help users find the word to type in the list of candidates by facilitating visual search in the layout of the graphical interface. Third, we provided word completion and compare whether word completion benefits TapGazer in terms of performance.

3.1 Virtual Keyboard Layout

Customization. TapGazer reuses the standard QWERTY layout to support learnability. However, in our pilot studies, we found people had varying finger preferences for typing on the QWERTY keyboard, e.g. key 'm' may be pressed with either the right index finger or the right middle finger. The mappings were consistent, i.e. remained overall stable for each user. As a result, TapGazer creates a profile for each user to record their finger-to-key mapping, also allowing users to map multiple fingers to the same letter (e.g. 'y' could be triggered by both the left and right index fingers). To guide novice users, we optionally visualize the customized finger-to-key mapping in a virtual keyboard and/or a hand model (Figure 2b), with each key colored according to its associated fingers and letters rendered on their corresponding fingers. Based on users' mappings, we generate prefix trees to quickly look up complete and incomplete candidate words and their word frequencies for each input string.

Feasibility. Text entry is only feasible if all the words in the homograph set of any input string can be somehow selected. The *minimum candidate number* (MCN) is the minimum number of candidate words the interface must be able to disambiguate at a time. It is equal to the maximum number of homographs an input string can have, i.e. it describes the worst possible ambiguity that may need to be resolved. The design needs to determine the MCN in advance because display space needs to be adequately allocated, or users must be given the option to page through sets of candidates. The MCN is also relevant for performance as it describes the worst-case scenario of visual search for the right candidate. We determined popular QWERTY-based finger-to-key mappings in pilot experiments and then ran a simulation to determine their overall MCN based on different word sources: the 1000 most common words ("1K") retrieved from Wikipedia with $MCN_{1K} = 4$; the standard MacKenzie phrase corpus [62], which contains 500 phrases for evaluation use, with $MCN_{MacKenzie} = 6$; and the 90% most frequent words (7,440, "7K") generated from the wordfreq library [1], which includes many very-low-frequency specialized words and acronyms that are not typically part of dictionaries, with $MCN_{7K} = 7$. We design our interface to be able to show at least 10 candidates to cover all English dictionary words and also many low-frequency non-dictionary

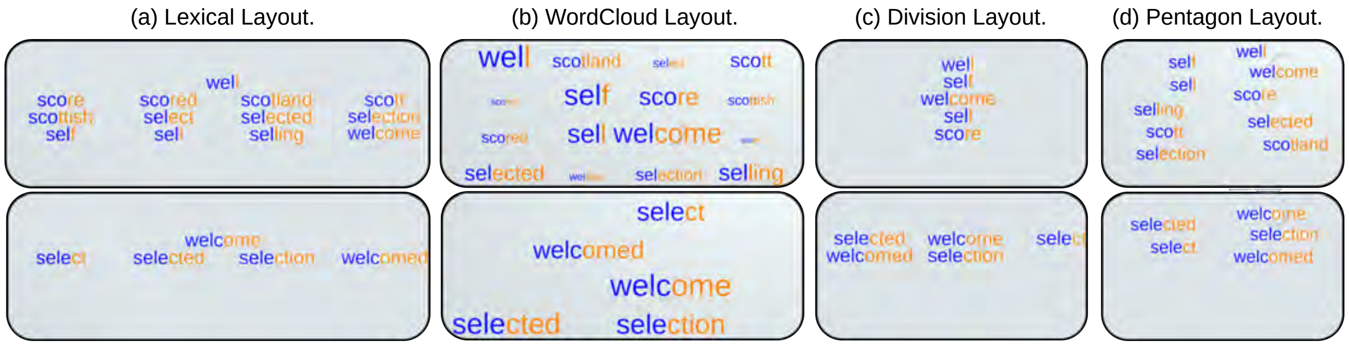


Figure 3: Evolution of TapGazer candidate layout designs: a) *Lexical Layout* places the most common candidate word in the first row and arranges the other candidates in alphabetical order. All the candidates have the same font size. b) *WordCloud Layout* emphasizes frequent candidates with a larger font size. Candidates that were already shown on the previous tap keep their position. c) *Division Layout* divides all candidates into three columns according to their last letter. d) *Pentagon Layout* orders the candidates based on the frequency and arranges the candidates in a compact single or double pentagon shape, separating them for easier gaze selection.

words across typical QWERTY finger-to-key mappings. For unsupported words such as neologisms and special acronyms, we provide a *spelling mode* for character-level text entry (see subsection 3.4).

Alternative Layouts. We also calculated the MCNs of standard keyboard layouts other than QWERTY to gauge their suitability for use in TapGazer. Optimal word gesture keyboards such as Smith et al.'s GK-D ($\text{MCN}_{\text{MacKenzie}} = 11$, $\text{MCN}_{7K} = 12$) and GK-T ($\text{MCN}_{\text{MacKenzie}} = 7$, $\text{MCN}_{7K} = 17$) [90] have higher MCNs, probably because they are not optimized for key-based typing. If the left thumb is used for tapping instead of deletion (e.g. by triggering deletion with a chord), having 9 fingers to tap reduces ambiguity in the finger-to-key assignment, potentially decreasing the MCN. We calculated the MCN for some known 9-key layouts: standard T9 [102] ($\text{MCN}_{\text{MacKenzie}} = \text{MCN}_{7K} = 5$); HiFinger [42], which distributes letters in lexical order over nine keys ($\text{MCN}_{\text{MacKenzie}} = 5$, $\text{MCN}_{7K} = 8$); and the quadmetric optimized layout [50] ($\text{MCN}_{\text{MacKenzie}} = \text{MCN}_{7K} = 4$). Finally, we performed an extensive combinatorial search of non-QWERTY layouts and found that there is a very large number of mappings for eight fingers with $\text{MCN}_{\text{MacKenzie}} = \text{MCN}_{7K} = 4$. These results suggest that layout optimization can help to reduce the number of candidates that have to be shown at one time, which could speed up text input.

3.2 Word Candidate Layout

The most important part of TapGazer's visual interface is the central gray area where word candidates are shown for selection by the user (Figure 2b). These candidates are colored to indicate the tapping progress of each word: the prefix of each word that has already been tapped is colored in blue, while yet-to-be-tapped postfixes are colored in orange. Complete candidates are completely blue and are always shown in the interface as they must be available as word choices. Any further available space can be filled with incomplete candidates, indicating options for word completion. The number of candidates shown is a trade-off between saving taps through word completion, and visual search time spent looking for the right candidate. Visual search time is affected by the way we arrange the candidates, therefore we designed, tested, and re-designed the layout to reach a suitable design. Figure 3 illustrates the design evolution of TapGazer's candidate layout.

Initial Design. We first designed (a) Lexical Layout and (b) WordCloud Layout based on the following design principles. *Systematic locations*: Users should intuitively know where to look for a word. *Salience*: More likely words should be more salient (e.g. larger or more central). *Continuity*: Avoiding changes in the position of a suggested word between taps may help users to spot it. Lexical Layout places the most frequent word into the first row by itself for salience, and fills the rows below with other candidates in alphabetical order to achieve systematic locations. This prioritizes systematic locations over continuity, as candidates' positions may change between taps, e.g. "welcome" in Figure 3. WordCloud Layout arranges candidates in word-cloud style, with more frequent words arranged at the center and in a larger font. Candidates keep their positions between taps, prioritizing continuity over systematic locations. Both layouts use only the central part of the VR display to avoid large eye movements.

Formative Design Study. To understand the effects of the layouts and their design principles on novices, we conducted a formative study with 12 participants (5 female, 7 male; aged 18 to 30, $M = 24.67$, $SD = 3.94$), comparing the two layouts which were implemented in Unity in a within-participant design. After a 5-minute training phase, each participant used each layout twice for 5 minutes each, with a small break in between, to enter random sentences from the MacKenzie corpus [62] as quickly and accurately as possible. Wearing a Tobii HTC VIVE Devkit gaze tracking VR headset, they tapped on a QWERTY keyboard, keeping their fingers on the same keys for tapping. To investigate the effects of a different input device, participants then repeated the task with the TapStrap using only their preferred layout. Each condition was followed by quantitative and qualitative questionnaires collecting their feedback on each layout, design principle and input device.

Paired t-tests showed no significant differences in typing speed ($t(11) = 0.897, p = .389$, Cohen's $d = 0.259$), accuracy in terms of Total Error Rate (TER) [91] ($t(11) = 0.099, p = .923, d = 0.029$), System Usability Scale (SUS) scores [4] ($t(11) = 1.081, p = .202, d = 0.312$), or NASA-TLX task load scores [37] ($t(11) = 1.307, p = .218, d = 377$). Participants were split equally in their layout preference. Their qualitative responses showed that they immediately understood Lexical layout's systematic locations but did not find them helpful in spotting target words quickly. Having the most frequent word at the top or center was found useful, but variations in font size were found to be distracting when typing low-frequency words ("words with larger font size draw too much attention and it became difficult to locate the infrequent words"). Some participants noticed WordCloud's continuity but did not find it beneficial ("confusing") as tapping was too fast to visually follow candidates.

In conclusion, our most important finding was that it is not useful to design the visual layout around the tapping process or around cognitively demanding criteria such as relative alphabetic ordering, as fingers are much faster than the eye or the mind [82]. It is more useful to consider the layout as a pure visual search task, where visual search time is correlated with the number of candidates and the distance of eye movement [72]. The study also highlights the importance of the tap input device: leaving the fingers on the same keys for tapping felt unnatural and slowed them down considerably (on average 15.37 WPM for Lexical and 14.34 WPM for WordCloud); participants liked the idea of TapStrap but were frustrated and slowed down by its low tap recognition rates (on average 9.89 WPM; TER of 0.34 vs. 0.15 for Lexical and 0.14 for WordCloud). We will introduce the input device we used for final evaluation in subsection 3.5.

Final Designs. Based on the formative study, we developed two new layouts that focus on optimizing visual search by reducing distances between words, improving salience of frequent words, dropping the continuity principle, and applying the systematic locations principle more carefully to avoid cognitive load: one layout for power users and one for novices. Division Layout (Figure 3c) distributes candidates into three columns according to their last letter, ordering each column by word frequency. The column boundaries were chosen to balance the expected number of candidates in each column, with words ending in A-E on the left, F-R in the middle, and S-Z on the right. This layout is designed for power users who have learned where to expect a word, potentially reducing search time by 2/3. The authors tested this on themselves over several days and found that with practice, the eyes would subliminally move to the right column when tapping frequent words. Pentagon Layout (d) is designed to be suitable both for novices and experts. It arranges candidates in compact groups of five, close together to minimize eye movement but with enough separation for accurate gaze selection (at least 0.5° visual angle between the edge of two neighbour words, which typically leads to considerably more separation between the center of any two words and enabled accurate selection in our pilot studies). The pentagon shapes mitigate overlap between long adjacent words and try to take advantage of people's ability to quickly scan groups of five items at a time [68]. Complete candidates are always shown before incomplete candidates, with frequent words closer to the top.

The two new layouts were delivered to a group of users remotely for subjective feedback. Most participants believed both layouts could facilitate fast typing given enough practice. However, they preferred the Pentagon layout because it was more compact (less "sprawling" and "confusing") and more straightforward and intuitive to search since they would usually scan for the word to type downwards one by one ("from the top"). Thus we chose Pentagon Layout for our main study as it is easier to use for non-experts.

3.3 Disambiguation

After presenting possible word candidates, users need to select a candidate to disambiguate the input. In text entry on mobile devices where word candidates are commonly selected by touch, users typically fixate on a candidate with their eyes right before and while selecting it [100], and similar gaze behaviour can be observed for pointer-based selection [8]. TapGazer takes advantage of these quick, subliminal fixations by employing gaze tracking for word selection to minimize taps and reduce cognitive load. Once the user has found the right word and is looking at it, the user can select it with a tap of the right thumb. We chose to use a tap rather than a gaze-dwell for selection as the latter is much more time-consuming and can lead to Midas Touch (inadvertent activations) [78]. We tested our gaze selection implementation With an HTC Vive Tobii DevKit for VR users and also a Tobii 5 tracker bar for non-VR users, showing a small transparent circle as gaze indicator to give users feedback about gaze tracking. Pilot user feedback showed that, based on the estimated gaze coordinates, it was possible to determined which candidate word was being gazed at.

In the absence of gaze tracking, we provide a variant of TapGazer with purely *manual selection* (Figure 4 bottom-right). In this variant, selecting a candidate is a two-step operation: 1) tapping with the right thumb, and 2) tapping with one of five fingers (right thumb, right middle, right index, left middle, left index) to select one among a maximum of five candidates shown. To support selection from more than five candidates, users can page through sets of five candidates with their left and right little fingers. The layout design helps to avoid paging operations by showing complete candidates first and ordering complete and incomplete candidates by their descending word frequencies.

Impact Study for Manual Selection. To understand how manual selection impacts TapGazer, we conducted a study with 20 participants (3 female and 17 male; aged 24 to 35, $M = 28.25$, $SD = 3.45$) using a within-participant design. The study was conducted remotely because of COVID restrictions, so participants used their personal computers without VR headset or gaze tracker. Participants were sent a Unity executable and completed the same procedure as in the formative design study (subsection 3.2) except that they were allowed to 'tap' using their whole keyboard (i.e. type as usual). We compared manual selection with simulated gaze selection, i.e. the prototype assumes the user gazed on the correct word when selecting a candidate. To mitigate the bias of simulated gaze, we impressed on our participants the importance of locating the right candidates with their eyes before selecting them with a key tap.

Paired t-tests showed that selection with simulated gaze was significantly faster than manual selection (average 51.84 vs. 36.85 WPM, $t(19) = 9.697, p < .001^{***}, d = 2.168$), with significantly higher SUS scores (77.00 vs. 60.63, $t(19) = 4.052, p < .001^{***}, d = 0.906$). The differences in TER (0.040 vs. 0.046, $t(19) = 1.422, p = .171, d = 0.318$) and TLX scores (36.94 vs. 54.14, t(19)=1.990, p=.061,

Workflow uist.pdf

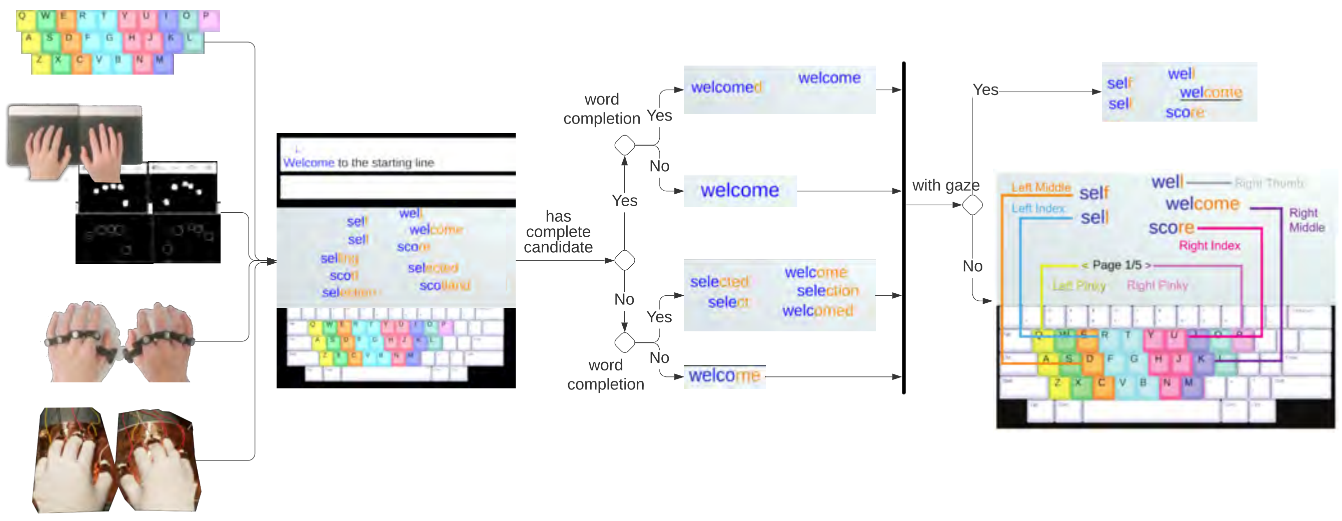


Figure 4: TapGazer's workflow: After receiving a tap from a suitable input device, TapGazer updates the candidates according to the word completion mode, and allows users to select a candidate either with gaze or with additional taps.

d=0.445) were not significant. Participants were able to reach 76% and 54% of their QWERTY typing speed, respectively. This setup favoured gaze selection because simulated gaze tracking does not suffer from tracking inaccuracy and selection mistakes, hence the results arguably estimate an upper bound for the impact of manual selection. While the reduction in performance is marked, the results indicate that manual selection is possible with a reasonable performance, and that users are able to learn it quickly.

3.4 Miscellaneous Text Entry Functionality

We have designed miscellaneous text editing functions for TapGazer in order to make it a complete text entry method. *Deletion* of the current input string is performed by tapping the left thumb, allowing users to start a word again. If the left thumb is pressed right after selecting a candidate, the candidates for the last input string will be shown again, allowing users to change the selection or tap the left thumb again to delete the word. *Spelling mode* is triggered with a chord operation. Users can switch between word-level and character-level text entry by tapping their left and right index fingers simultaneously. Afterward, users can rotate through the characters mapped to each finger by repeatedly tapping a respective finger, and enter the character by tapping the right thumb. Tapping the right thumb again concludes the character-level input. *Cursor navigation* with gaze is performed by selecting words in the entered text directly with gaze and right thumb [89], or by entering a cursor navigation mode through a button in the periphery of the interface [59] with gaze and right thumb. Users can then move the cursor by tapping the left/right index finger and exit cursor mode with a right thumb tap. If the gaze is unavailable, users can enter cursor mode by tapping the right index and ring fingers simultaneously.

3.5 Input Devices

We tested TapGazer with several off-the-shelf input devices (Figure 4 left): 1) QWERTY keyboards are partitioned into areas that are each mapped to one finger. This partitioning is consistent with a user's usual finger-to-key mapping, so the user can retain their QWERTY skills. 2) Touchpads (Sensel Morph in our case) report pressure images of fingers for every frame. We detect the hand directions (left and right hand) and identify fingers based on the shape and configuration of recent pressure points. Users can calibrate the finger detection at any time by placing all fingers onto the touchpad. In pilot studies we estimated the accuracy of finger detection on Sensel Morph touchpads at 99.86%. 3) Wearable devices such as TapStrap can report tapping information through Bluetooth. In addition to TapStrap, which had a comparatively poor accuracy (see subsection 3.2), we also designed a pair of touch-sensitive gloves that report taps with finger identities (Figure 4 bottom left). We connected a pair of cotton gloves to an Arduino UNO board through wires with conductive foil tape around each finger, and used foil tape on hard and soft surfaces such as tabletops and things to detect taps based on electric currents. In pilot studies we estimated the accuracy of the gloves at close to 100%.

3.6 Discussion

Our current prototype has limitations. Word prediction is based only on word frequency; it could be improved by also taking the context of a word into account. We also do not provide auto-correction (neither at the character level of finger-key mappings nor the word level), as this could confound our study of accuracy. Finally, there is the wider design question of using only the finger identities of taps: is it a good idea to disregard finger positions altogether, although they were shown to be beneficial to input speed when tapping on hard, flat surfaces [84]? Most participants of the pilot studies presented in this section believed that just tapping could be more efficient, and there is some evidence in session input logs (from the study in subsection 3.3) of users becoming 'lazy' and just tapping the correct finger instead of hitting the corresponding QWERTY key. Our main motivation was that "just tapping" would better support the use of non-standard surfaces and poses such as one's thighs while standing, which will be further investigated in the next section. TapGazer was not designed to replace typing, but rather to complement it to address casual text entry in VR.

4 EVALUATION

We conducted two user studies to examine I) how the different TapGazer variants perform in terms of speed, accuracy and user preference (RQ2), as compared to typing on a physical QWERTY keyboard outside of VR, and II) how tapping on the thighs in different poses (sitting vs. standing) impacts TapGazer performance. In Evaluation I, we used a within-participant design with three conditions: typing on a standard QWERTY keyboard (K) in a typical manner outside of VR, using TapGazer with gaze candidate selection and word completion (GC) while wearing a VR headset, and using TapGazer with gaze candidate selection and no word completion (GN) while wearing a VR headset. Participants were sitting at a desk in all conditions and were tapping on touchpads (two Sensel Morphs) when using TapGazer. In Evaluation II we used a within-participant design with three conditions: typing on a standard QWERTY keyboard (K) outside of VR, tapping on thighs while sitting (SIT) in VR, and tapping on thighs while standing (STAND) in VR with a pair of touch-sensitive gloves made by us. The order of the conditions was counterbalanced to mitigate the effects of learning and fatigue. All TapGazer prototypes were built in Unity, using the Pentagon layout. Participants wore a Tobii HTC VIVE Devkit HMD with gaze tracking connected to a windows laptop (Intel Core i7, NVidia GeForce RTX 2070) in the TapGazer conditions.

Measures. We compared the conditions using as objective performance measures typing speed (WPM) and total error rate (TER) [91], and as subjective measures SUS usability scores [4] and NASA-TLX task load scores [37]. Each tap/keystroke operation was recorded for analysis. WPM and TER were calculated following Mackenzie et al.'s definition [91]: $WPM = \frac{|T-1|}{S} \times 60/5$, where $|T|$ is the length of the final transcribed string and S is the elapsed time in seconds from the first to the last tap in a phrase; $TER = \frac{INF+IF}{C+INF+IF}$, where INF is the number of incorrect keystrokes that were not fixed by the user, IF are keystrokes (excluding editing keys) that occurred in the input stream but not in the transcribed text, and C are correct keystrokes.

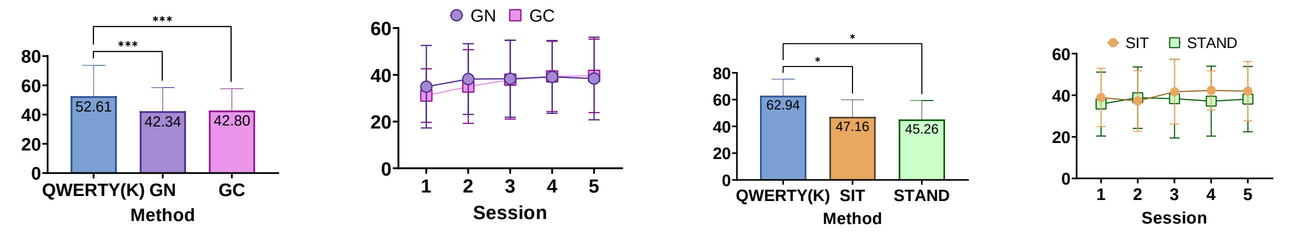
Procedure. Both Evaluation I and II followed a similar procedure. After a brief introduction of TapGazer, participants were asked to type phrases randomly selected from the MacKenzie & Soukoreff corpus [62] on a standard QWERTY keyboard, to measure their speed and TER. Then we recorded their finger-to-key mapping preference: participants were instructed to first type letters from 'A' to 'Z' and then random phrases from the MacKenzie & Soukoreff corpus [62], while being observed by the experimenter. Customized finger-to-key mappings for each participant were then generated with a Python script according to the fingers they were using for each letter. Participants were able to update their mappings during the following training sessions if desired. For both evaluations, they performed each of the two conditions (GC vs. GN for Evaluation I, and SIT vs. STAND for Evaluation II) in counterbalanced order, with each condition consisting of a training session and five test sessions. First, gaze calibration was performed after putting on the VR headset. Then, in the training session, participants were given a brief tutorial of the respective text entry condition and were able to practice it for a couple of minutes until they felt comfortable. In the following five test sessions, participants were again asked to enter phrases randomly sampled from the MacKenzie & Soukoreff corpus [62] as fast and accurately as possible. Each test session was one minute long and participants were allowed to take short breaks between the sessions. In Evaluation I, participants completed SUS and NASA-TLX questionnaires after each condition. Each condition took around 10-15 minutes. After finishing all two conditions, participants completed a demographics questionnaire and for Evaluation I, ranked the conditions according to their preference. Lastly, participants were interviewed about their TapGazer experience. Each experiment took between 40 and 60 minutes.

Participants. For Evaluation I, we recruited 14 participants (10 male, 4 female), with an average age of 23.6 (SD = 1.7). All of them used a QWERTY keyboard regularly. Their QWERTY typing speed was on average 52.61 WPM (SD = 21.07) with a TER of 11.5% (SD = 5.85%). 11 had used eye-tracking devices before. For Evaluation II, we recruited 5 different participants (3 male, 2 female), with an average age of 23.8 (SD = 1.1). Their QWERTY typing speed was on average 62.94 WPM (SD = 12.38) with a TER of 9.3% (SD = 3.3%). 4 had used eye-tracking devices before.

4.1 Results

We validated that the data satisfies the assumptions of a repeated measures analysis of variance (ANOVA). We used one-way repeated measures ANOVAs to compare effects across all TapGazer conditions, and two-way repeated measures ANOVAs to compare the effects of TapGazer variants with regards to the factors Completion (word completion vs. no word completion) and Session in Evaluation I, and Pose (sitting vs. standing) and Session in Evaluation II. Paired t-tests with Holm correction were used for all pairwise comparisons between conditions. All tests for significance were made at the $\alpha = 0.05$ level. The error bars in the graphs show the standard error.

Text Entry Speed. In Evaluation I (Figure 5a), users tapped at M = 42.80 WPM (SD = 14.87) for GC, M = 42.34 WPM (SD = 16.11) for GN, and M = 44.81 WPM (SD = 14.67) for their preferred TapGazer variant. The main effect of Condition (K, GN, GC) ($F(2, 26) = 11.15, p < .001^{***}$) was significant. K was significantly faster than both TapGazer variants ($t(13) \geq 3.93, p < .002^{**}$). The main effect of Session (1 to 5) ($F(4, 52) = 13.08, p < .001^{***}$) was significant, while the main effect of Completion (GN, GC) ($F(1, 13) = 0.42, p = .53$) and the interaction effect of Completion and Session ($F(4, 52) = 1.16, p = .34$) were not significant (Figure 5b). In Evaluation II (Figure 5c), users tapped at M = 47.16 WPM (SD = 12.74) for SIT and M = 45.26 WPM (SD = 14.12) for STAND. The main effect of Condition (K, SIT, STAND) ($F(2, 8) = 7.76, p = .013^{*}$) was significant. K was significantly faster than SIT ($t(8) = 3.2, p = .037^{**}$) and STAND ($t(8) = 3.6, p = .021^{**}$).



(a) Eval I: WPM of QWERTY, GN and GC. **(b) Eval I: WPM of GN and GC across 5 sessions.** **(c) Eval II: WPM of QWERTY, SIT and STAND.** **(d) Eval II: WPM of SIT and STAND across 5 sessions.**

Figure 5: Evaluation results for WPM comparing QWERTY keyboard (K), TapGazer with gaze selection and no word completion (GN), and TapGazer with gaze selection and word completion (GC), as well as SIT and STAND poses while tapping on the thighs.

The main effects of Pose (SIT, STAND) ($F(1,4) = 1.32, p = .33$) and Session ($F(4,16) = 0.40, p = .81$), and the interaction effect ($F(4,16) = 0.30, p = .88$) were not significant (Figure 5d). Users saved on average 3.4% of taps by using word completion in GN (if there was only one possible candidate) and 7.1% of taps in GC. Using word completion in GC could theoretically have saved up to 34.2%, but participants used word completion mostly only for long words.

Error Rate. In Evaluation I (Figure 6a), users typed with a TER of $M = 11.55\%$ ($SD = 0.058$) for K, $M = 2.13\%$ ($SD = 0.017$) for GN, and $M = 3.64\%$ ($SD = 0.026$) for GC. The main effect of Condition ($F(1,13) = 12.62, p = .004^{**}$) was significant, while the main effect of Session ($F(4,52) = 1.44, p = .23$) and the interaction effect of Condition and Session ($F(4,52) = 1.21, p = .38$) were not significant. All TapGazer variants had significantly lower error rates than K ($t(13) \geq 6.53, p < .001^{***}$). In Evaluation II (Figure 6a), users typed with a TER of $M = 10.25\%$ ($SD = 0.039$) for K, $M = 3.69\%$ ($SD = 0.029$) for SIT, and $M = 4.03\%$ ($SD = 0.028$) for STAND. The main effects of Pose ($F(1,4) = 0.97, p = .40$) and Session ($F(4,16) = 0.56, p = .70$), and the interaction effect ($F(4,16) = 0.3, p = .87$) were not significant. Again, all TapGazer variants had significantly lower error rates than K ($t(4) \geq 6.47, p < .001^{***}$).

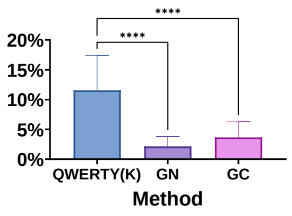
Usability and Workload. Figure 6c shows the average SUS usability scores for Evaluation I, with $M = 69.64$ ($SD = 18.55$) for GN and $M = 73.21$ ($SD = 11.87$) for GC. Figure 6d shows the average NASA-TLX task load scores, with $M = 48.47$ ($SD = 11.15$) for GN and $M = 48.60$ ($SD = 13.49$) for GC. The differences in SUS scores ($t(13) = -0.109, p = .29$) and NASA-TLX scores ($t(13) = 0.378, p = .71$) between GN and GC were not significant.

Preferences and Qualitative Feedback. When asked about which variant of Tapgazer they preferred in Evaluation I, 10 of the 14 participants preferred GC and 4 preferred GN. In the post-interviews, participants were overall positive about Tapgazer ("*I can type very fast after practice*", "*save energy by just tapping without reaching the specific letter*"). Several participants stated it was easy to find the right candidate words ("*the candidate words are different from each other and easy to locate the word to type*", "*look at where the word will show up and select it when it shows*"). Most participants appreciated the ability to complete words ("*I'd love to have more words to select from*", "*can save quite a few keystrokes when using TapGazer with completion*", "*I want to scan all candidates words in case I found the correct one*"), but some noted that it was easier not to consider completion of words ("*focus on typing the word and no need to worry about the candidates shown. And usually I don't need to type the entire word when it is long*" – referring to the fact that without word completion, TapGazer shows the most likely incomplete candidate if there are no complete candidates, allowing users to quickly complete long words). In Evaluation II, all participants reported that they would be willing to use TapGazer for off-desktop scenarios like VR as it felt comfortable in both a seated and standing pose.

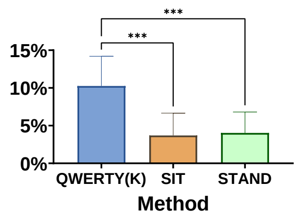
4.2 Discussion

Our results demonstrate that overall, TapGazer is an easy-to-use system that can reach similar or higher average typing speeds than other text entry methods addressing similar use cases (see Table 1): on average 47.2 WPM when tapping on the thighs while sitting, compared to 44.6 WPM and 41.0 WPM, respectively, for the best competitors TOAST [88] and VelociTap [96]. In contrast to most competitors, Evaluation II demonstrates that TapGazer can be used effectively by tapping on one's thighs, both in a sitting and a standing pose, without apparent loss of performance. Furthermore, TapGazer achieves significantly lower error rates than the QWERTY keyboard, as word-level text entry avoids some sources of error of character-level text entry: while QWERTY typing, participants frequently used the correct finger on the wrong key – a mistake that does not affect TapGazer. Our study data logs show that mistakes due to inconsistent finger-to-key mappings happened very rarely in TapGazer. The higher WPM averages listed in Table 1 are mainly for experienced 'expert' users; however, our participants were all novice users of TapGazer, so it is plausible that even better performance could be achieved with more practice. TapGazer with and without word completion perform similarly and both have their place: some users prefer to just type and not think about the completion of words, while others prefer to look for incomplete candidates before completion. We analyze these two strategies further in subsection 5.1.

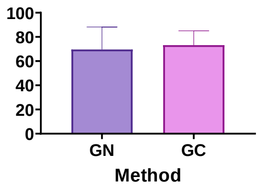
When comparing TapGazer with the best alternative method, TOAST [88], it appears that TOAST is potentially more efficient at entering individual letters: it uses a standard QWERTY layout on a



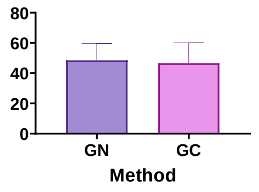
(a) Eval I: TER of QWERTY, GN, and GC.



(b) Eval II: TER of QWERTY, SIT, and STAND.



(c) Eval I: SUS scores of GN and GC.



(d) Eval I: NASA-TLX scores of GN and GC.

Figure 6: On the left, evaluation results for TER comparing a) QWERTY keyboard (K), TapGazer with gaze selection and no word completion (GN), and TapGazer with gaze selection and word completion (GC), as well as b) SIT and STAND poses while typing on thighs. On the right, evaluation results for c) SUS and d) TLX comparing GN with GC.

table-mounted touchpad where specific key strokes are detected, as opposed to just finger taps. However, TOAST requires additional key strokes to select candidate words and avoid input errors, which is likely less efficient than TapGazer's gaze selection. The second-best alternative method, VelociTap [96], uses finger taps on a small mobile display to enter letters; this is likely slower than TapGazer's tapping, which resembles normal QWERTY typing. However, VelociTap appears to use a very efficient sentence-based language model for text completion, which is more sophisticated than the model we were using for TapGazer.

Limitations. Our study used novices and is mainly cross-sectional, thereby likely underestimating the average performance of longer-term users. Both evaluations had fairly small sample sizes; however, they were able to detect the differences in WPM and TER between TapGazer and QWERTY keyboard. The differences between the TapGazer conditions (GN vs. GC and SIT vs. STAND) had only small effect sizes, therefore it is unlikely that the statistical test results would change with more participants [18]. The results of Evaluation II are roughly in keeping with those of Evaluation I, supporting their validity.

The use of gaze in TapGazer is not only a main strength but likely also its biggest limitation: besides typical challenges of gaze tracking such as availability and accuracy, users of TapGazer need to use their gaze while typing, which makes eyes-free typing infeasible. Users cannot simultaneously look at other things, such as (avatars of) other users or visual content to transcribe. Furthermore, finger tap detection poses its own challenges and will likely require some kind of extra hardware such as gloves in many cases.

5 MODEL-BASED ANALYSIS

In the following we will discuss models describing the user performance of TapGazer and its variants (RQ3) and then apply them to the analysis of design options and usage strategies. Because TapGazer is based on QWERTY typing, it is plausible to estimate performance based on a user's QWERTY typing speed. Based on the data from Section 4, WPM_K is significantly positively correlated with WPM_{GN} ($r(14) = 0.88, p < .001^{***}$) and with WPM_{GC} ($r(14) = 0.87, p < .001^{***}$), with 'large' effect sizes. Linear slope regression analysis yielded significant regression equations: $WPM_{GN} = 0.788 \times WPM_K$ and $WPM_{GC} = 0.792 \times WPM_K$. That is, users achieved on average 79% of their QWERTY typing speed when using the GC variant. Similarly, the average time taken for typing a key on a QWERTY keyboard $type_K$ is significantly positively correlated with the average times for tapping a finger tap_{GN} ($r(14) = 0.91, p < .001^{***}$) and tap_{GC} ($r(14) = 0.82, p < .001^{***}$). Linear slope regression analysis yielded significant regression equations: $tap_{GN} = 1.28 \times type_K$ and $tap_{GC} = 1.30 \times type_K$. Such regression analyses confirm that there is a strong linear relationship between QWERTY typing and TapGazer performance with novice users, across various input devices, TapGazer variants and poses. This makes QWERTY typing speed a good estimator of TapGazer performance and a good covariate to consider for fair comparisons of TapGazer variants between different users.

5.1 Slow Typists

Some people are slow typists, e.g. when they are just learning to type. Word completion can be particularly useful for them. This is similar to text entry on a mobile phone, where tapping individual keys can be slow and many people use word completion extensively to speed up text input. In the following we show how to estimate the QWERTY typing speed that marks the point in typing and tapping skill where not using word completion becomes faster than using word completion with a visual search for the correct word after every tap.

Similar to the well-known Keystroke-Level Model (KLM) [14], we model the time T_{GN} required for entering a word w in TapGazer without word completion based on: a) the average tapping time for fingers *tap*; b) the average tapping time for the right thumb *space* (which types space in the standard QWERTY mapping), and c) the average visual search time $search_{GN}$ for finding a desired word among the completed words shown: $T_{GN}(w) = |w| \times tap + search_{GN} + space$. That is, we sum up the average tapping time for each of the $|w|$ letters, the average search time for spotting the right completed word, and the average time of the confirmatory tap with the thumb. Note that by definition, this model predicts the average word completion time for our evaluation of GN exactly when substituting our measured average values for the model parameters. Similarly, we model the time T_{GC} required for entering a word w in TapGazer with word completion, assuming that the user looks at the suggested words after every tap. This time we consider the number of taps $c(w) \leq |w|$ required until w appears in the list of suggestions, and the average visual search time $search_{GC}$ for finding a desired word among suggested, possibly incomplete words: $T_{GC}(w) = c(w) \times (tap + search_{GC}) + space$. The model illustrates the trade-off between a reduced number of taps and increased time spent per tap.

In Section 5 we have shown that there is a strong linear relationship between tapping and QWERTY typing speed. Therefore, in order to estimate T_{GN} based on the time T_K required to type word w, we substitute *tap* and *space* by corresponding linear estimates $1.28 \times type_K$ and $0.94 \times type_K$, respectively. Because search times do not vary with QWERTY speed, we approximate them using averages $search_{GN} = 270$ ms and $search_{GC} = 331$ ms. The latter is the average for GC when the maximum number of candidates is shown (10) so that it is not immediately apparent which candidate to choose, as this is the most likely case when looking for word completions after every tap. Furthermore, we substitute the average word length in English text $|w| = 4.79$ [71], and the expected number of taps $c = 2.41$ required before a desired word appears in the suggestions. The latter was determined using a simulation of the word-frequency based suggestion algorithm used in GC on a dictionary of the 7,582 most frequent English words. This results in estimates of the average times per word dependent on $type_K$: $T_{GN} = 7.05 \times type_K + 270$ ms and $T_{GC} = 3.95 \times type_K + 797.71$ ms. T_{GN} and T_{GC} are equal at $type_K = 170$ ms, which is equivalent to about $WPM_K = 60.95$. Therefore, typists much slower than that would likely be faster using TapGazer with word completion. A better word prediction algorithm will reduce the expected value for $c(w)$, increasing the estimated speed at which word completion becomes a hindrance. A similar analysis can be done for the non-gaze variants of TapGazer.

5.2 Power Users

If the prediction algorithm used to generate suggestions for word completion is reasonably stable, i.e. if users can anticipate which word will be suggested as the most likely option, then power users will learn for frequent words how many taps they need before they can simply accept the most likely suggested word. In both GC and MC, the most likely suggestion can be quickly accepted without even looking at the word suggestions, by tapping the right thumb. Let us assume a power user has learned all the prefixes that must be tapped to make each of the 100 most frequent words of the English language the most likely suggestion, e.g. "tapping 't' makes 'the' the most likely word." According to our word frequency data, the 100 most common words account for 47.07% of all English texts. Let $c(w)$ be the number of taps a user needs to do before the word suggested as most likely is the desired word w. Similar to Section 5.1, this leads to the following model for a power user who uses word completion without visual search for the 100 most frequent words (first summand) and types words in full otherwise (second summand, using $search_{GN}$ as the search is only among the completed words, which come first): $T_{GC}(w) = 47.07\%(c(w) \times tap + space) + 52.93\%(|w| \times tap + search_{GN} + space)$.

According to our simulation of the word-frequency based suggestion algorithm used in GC and MC, which is based on the 7,582 most frequent English words, the expected number of taps a user needs to make before one of the 100 most frequent words becomes the most likely suggestion is $c = 2.05$. This is lower than one might think, as the three most frequent words (the, of, and), which account for more than 14% of English texts, all use different fingers on their first tap, so each appears immediately as a most likely suggestion. Furthermore, our simulation reveals that six of the 100 most frequent words (my, or, if, now, our, then, go) are never shown as most likely suggestion; they typically make up 1.15% of English texts, therefore we shift this percentage from the first to the second summand in our model. As in Section 5.1, we substitute c, the average word length in English texts $|w| = 4.79$, and estimates of $search_{GN}$, *tap* and *space*. In order to relate the model to QWERTY typing speed, we describe *tap* and *space* as linear estimates of $type_K$. The result is $T_{GC} = 5.49 \times type_K + 143$; the corresponding WPM_{GC} can be approximated for typical QWERTY typing speeds (up to 80 WPM) with a linear lower bound as $WPM_{GC} = 0.88 \times WPM_K$ (compared to $0.79 \times WPM_K$ for novice users). That is, by learning tap prefixes for frequent words so that these words can be selected quickly without visual search, TapGazer is expected to allow power users to achieve typing speed closer to QWERTY typing. Even if a user learns tap prefixes only for the 10 most common words, this accounts for about 22.22% of English texts.

When using gaze tracking, if a power user furthermore learns where a frequent word appears for the first time in the suggestions, e.g. "after tapping the left ring finger 'with' appears at the centre left", then the power user could potentially look at the right suggestion and select it immediately, reducing the average number of taps per word c and leading to an estimate of $WPM_{GC} = 1.03 \times WPM_K$ for the 100 and $WPM_{GC} = 0.84 \times WPM_K$ for the 10 most frequent words. If a power user is willing to learn a new layout, i.e. a finger-to-letter mapping not based on QWERTY, then c can be reduced further. We used branch-and-bound search to find a mapping that minimises c for the 100 frequent words, resulting in a mapping with $c = 1.18$ and $WPM_{GC} = 1.03 \times WPM_K$ if the positions of the respective word suggestions are also learned. In summary, learning tap prefixes and even display positions for common words can potentially speed TapGazer up drastically, with and without gaze tracking.

5.3 Discussion

Similar to KLM [14], our models are based on the average measurements obtained from our evaluation. As a result, their predictions will be inaccurate to to some degree when applied to different groups of users. In particular, our experiments collected TapGazer performance data only from novice users, and it is likely that users will get faster with practice. The models we created are therefore likely to underestimate the performance of longer-term users, forming a reference baseline for future research. Also, the models add value by formalising strategies that some users will likely apply to increase their TapGazer performance. Finally, the models identify important parameters affecting TapGazer's performance, providing starting points for further improvements in future work.

6 CONCLUSION AND FUTURE WORK

We have presented TapGazer, a novel text entry method combining tapping and gaze. TapGazer was designed to facilitate casual text entry in VR, without the need for a physical keyboard. Our results indicate that novice users can achieve 79% of their QWERTY typing speed, with an average TapGazer WPM of 44.81, which surpasses the performance of comparable text entry methods. Furthermore, the error rate of TapGazer is significantly lower than for a physical QWERTY keyboard, and TapGazer can be used in different poses (sitting and standing) while tapping on one's thighs without marked loss of performance. We have created performance models illustrating how different users can benefit from different usage strategies, and identifying performance parameters that can be optimized in future design iterations.

In the future, when affordable AR glasses with gaze and finger tracking will be as ubiquitous as mobile phones are today, wearers of those glasses may use TapGazer for text entry, e.g. by tapping on their thighs while waiting at a bus stop or walking down the street. This is an exciting direction for future work.

ACKNOWLEDGMENTS

This research was supported by NSF grant 1626098 and a grant from the Verizon Corporation. We thank all our study participants.

REFERENCES

[1] 2. LuminosoInsight/wordfreq: V2.2. https://doi.org/10.5281/zenodo.1443582

[2] Jiban Adhikary. 2018. Text Entry in VR and Introducing Speech and Gestures in VR Text Entry. In *MobileHCI*. Association for Computing Machinery, Barcelona, Spain, 1083–1092. https://doi.org/10.20870/IJVR.2019.19.3.2917

[3] Sunggeun Ahn and Geehyuk Lee. 2019. Gaze-Assisted Typing for Smart Glasses. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 857–869. https://doi.org/10.1145/3332165.3347883

[4] Aaron Bangor, Philip T Kortum, and James T Miller. 2008. An Empirical Evaluation of the System Usability Scale. *Intl. Journal of Human-Computer Interaction* 24, 6 (2008), 574–594. https://doi.org/10.1080/10447310802205776

[5] Nikolaus Bee and Elisabeth André. 2008. Writing With Your Eye: a Dwell Time Free Writing System Adapted to the Nature of Human Eye Gaze. In *International Tutorial and Research Workshop on Perception and Interactive Technologies for Speech-Based Systems*. Springer, Springer, 111–122. https://doi.org/10.1007/978

[6] Burak Benligiray, Cihan Topal, and Cuneyt Akinlar. 2019. SliceType: Fast Gaze Typing With a Merging Keyboard. *Journal on Multimodal User Interfaces* 13, 4 (2019), 321–334. https://doi.org/10.1007/s12193-018-0285-z

[7] Xiaojun Bi, Barton A Smith, and Shumin Zhai. 2010. Quasi-Qwerty Soft Keyboard Optimization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 283–286. https://doi.org/pdf/10.1145/1753326.1753367

[8] Hans-Joachim Bieg, Lewis L Chuang, Roland W Fleming, Harald Reiterer, and Heinrich H Bülthoff. 2010. Eye and pointer coordination in search and selection tasks. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*. 89–92.

[9] Gaddi Blumrosen, Katsuyuki Sakuma, John Jeremy Rice, and John Knickerbocker. 2020. Back to Finger-Writing: Fingertip Writing Technology Based on Pressure Sensing. *IEEE Access* 8 (2020), 35455–35468. https://doi.org/10.1109/ACCESS.2020.2973378

[10] Costas Boletsis and Stian Kongsvik. 2019. Controller-Based Text-Input Techniques for Virtual Reality: an Empirical Comparison. *International Journal of Virtual Reality* 19, 3 (2019), 2–15.

[11] Sidney Bovet, Aidan Kehoe, Katie Crowley, Noirin Curran, Mario Gutierrez, Mathieu Meisser, Damien O Sullivan, and Thomas Rouvinez. 2018. Using Traditional Keyboards in VR: SteamVR Developer Kit and Pilot Game User Study. In *2018 IEEE Games, Entertainment, Media Conference (GEM)*. IEEE, IEEE, 1–9. https://doi.org/10.1109/GEM.2018.8516449

[12] Doug Bowman, Vinh Ly, Joshua Campbell, and Virginia Tech. 2001. Pinch Keyboard: Natural Text Input for Immersive Virtual Environments. (01 2001). https://doi.org/10.1007/978-3-642-24082-_94

[13] Damien Brun, Charles Gouin-Vallerand, and Sébastien George. 2019. Keycube Is a Kind of Keyboard (k3). In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–4. https://doi.org/fullHtml/10.1145/3290607.3313258

[14] Stuart K Card, Thomas P Moran, and Allen Newell. 1980. The Keystroke-Level Model for User Performance Time With Interactive Systems. *Commun. ACM* 23, 7 (1980), 396–410. https://doi.org/10.1145/358886.358895

[15] Steven J Castellucci, I Scott MacKenzie, Mudit Misra, Laxmi Pandey, and Ahmed Sabbir Arif. 2019. TiltWriter: Design and Evaluation of a No-Touch Tilt-Based Text Entry Method for Handheld Devices. In *Proceedings of the 18th International Conference on Mobile and Ubiquitous Multimedia*. 1–8. https://doi.org/10.1145/3365610.3365629

[16] Morokot Cheat and Manop Wongsaisuwan. 2018. Eye-Swipe Typing Using Integration of Dwell-Time and Dwell-Free Method. IEEE, IEEE, 205–208. https://doi.org/10.1109/ECTICon.2018.8619868

[17] Sibo Chen, Junce Wang, Santiago Guerra, Neha Mittal, and Soravis Prakkamakul. 2019. Exploring Word-Gesture Text Entry Techniques in Virtual Reality. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–6. https://doi.org/10.1145/3290607.3312762

[18] Jacob Cohen. 1992. A Power Primer. *Psychological Bulletin* 112, 1 (1992), 155. https://doi.org/70039-power-primer-summary-article-cohen-1992

[19] Gennaro Costagliola, Vittorio Fuccella, and Michele Di Capua. 2011. Text Entry With Keyscretch. In *Proceedings of the 16th International Conference on Intelligent User Interfaces*. 277–286. https://doi.org/10.1145/1943403.1943445

[20] Wenzhe Cui, Suwen Zhu, Mingrui Ray Zhang, H Andrew Schwartz, Jacob O Wobbrock, and Xiaojun Bi. 2020. JustCorrect: Intelligent Post Hoc Text Correction Techniques on Smartphones. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 487–499. https://doi.org/10.1145/3379337.3415857

[21] Seyed Amir Ahmad Didehkhorshid, Siju Philip, Elaheh Samimi, and Robert J Teather. 2020. Text Input in Virtual Reality Using a Tracked Drawing Tablet. In *International Conference on Human-Computer Interaction*. Springer, 314–329.

[22] Tafadzwa Joseph Dube and Ahmed Sabbir Arif. 2019. Text Entry in Virtual Reality: a Comprehensive Review of the Literature. In *International Conference on Human-Computer Interaction*. Springer, Springer, 419–437. https://doi.org/10.1145/3359996.3364265

[23] Tafadzwa Joseph Dube and Ahmed Sabbir Arif. 2020. Impact of Key Shape and Dimension on Text Entry in Virtual Reality. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–10. https://doi.org/pdf/10.1145/3334480.3382882

[24] John J Dudley, Keith Vertanen, and Per Ola Kristensson. 2018. Fast and Precise Touch-Based Text Entry for Head-Mounted Augmented Reality With Variable Occlusion. *ACM Transactions on Computer-Human Interaction (TOCHI)* 25, 6 (2018), 1–40. https://doi.org/10.1145/3232163

[25] Mark D Dunlop, Naveen Durga, Sunil Motaparti, Prima Dona, and Varun Medapuram. 2012. QWERTH: an Optimized Semi-Ambiguous Keyboard Design. In *Proceedings of the 14th International Conference on Human-Computer Interaction With Mobile Devices and Services Companion*. 23–28. https://doi.org/pdf/10.1145/2371664.2371671

[26] Francine Evans, Steven Skiena, and Amitabh Varshney. 1999. VType: Entering Text in a Virtual World. *Submitted to International Journal of Human-Computer Studies* (1999). https://doi.org/10.1145/1044588.1044662

[27] Jacqui Fashimpaur, Kenrick Kin, and Matt Longest. 2020. PinchType: Text Entry for Virtual and Augmented Reality Using Comfortable Thumb to Fingertip Pinches. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems Extended Abstracts*. 1–7. https://doi.org/10.1145/3334480.3382888

[28] John M Findlay. 1997. Saccade Target Selection During Visual Search. *Vision Research* 37, 5 (1997), 617–631.

[29] Yulia Gizatdinova, Oleg Špakov, and Veikko Surakka. 2012. Comparison of Video-Based Pointing and Selection Techniques for Hands-Free Text Entry. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*. 132–139. https://doi.org/pdf/10.1145/2254556.2254582

[30] Jun Gong, Zheer Xu, Qifan Guo, Teddy Seyed, Xiang'Anthony' Chen, Xiaojun Bi, and Xing-Dong Yang. 2018. Wristext: One-Handed Text Entry on Smartwatch Using Wrist Gestures. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–14. https://doi.org/10.1145/3173574.3173755

[31] Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. 2002. Language Modeling for Soft Keyboards. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*. 194–195. https://doi.org/10.1145/502716.502753

[32] Nathan Green, Jan Kruger, Chirag Faldu, and Robert St. Amant. 2004. A Reduced QWERTY Keyboard for Mobile Text Entry. In *CHI'04 Extended Abstracts on Human Factors in Computing Systems*. 1429–1432. https://doi.org/pdf/10.1145/985921.986082

[33] Jens Grubert, Lukas Witzani, Eyal Ofek, Michel Pahud, Matthias Kranz, and Per Ola Kristensson. 2018. Text Entry in Immersive Head-Mounted Display-Based Virtual Reality Using Standard Keyboards. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, IEEE, 159–166. https://doi.org/10.1109/VR.2018.8446059

[34] Jan Gugenheimer, David Dobbelstein, Christian Winkler, Gabriel Haas, and Enrico Rukzio. 2016. Facetouch: Enabling Touch Interaction in Display Fixed Uis for Mobile Virtual Reality. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 49–60. https://doi.org/10.1145/2984511.

2984576
[35] Aakar Gupta, Cheng Ji, Hui-Shyong Yeo, Aaron Quigley, and Daniel Vogel. 2019. RotoSwype: Word-Gesture Typing Using a Ring. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12. https://doi.org/10.1145/3290605.3300244
[36] Shangchen Han, Beibei Liu, Robert Wang, Yuting Ye, Christopher D Twigg, and Kenrick Kin. 2018. Online Optical Marker-Based Hand Tracking With Deep Labels. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–10. https://doi.org/10.1145/3197517.3201399
[37] Sandra G Hart. 2006. NASA-Task Load Index (NASA-TLX); 20 Years Later. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 50. Sage publications Sage CA: Los Angeles, CA, Sage publications Sage CA: Los Angeles, CA, 904–908.
[38] Anke Huckauf and Mario H Urbina. 2008. Gazing With PEYEs: Towards a Universal Input for Various Applications. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications*. 51–54. https://doi.org/10.1145/1344471.1344483
[39] Robert JK Jacob. 1993. Eye Movement-Based Human-Computer Interaction Techniques: Toward Non-Command Interfaces. *Advances in Human-Computer Interaction* 4 (1993), 151–190. https://doi.org/10.1145/332040.332445
[40] Haiyan Jiang and Dongdong Weng. 2020. HiPad: Text Entry for Head-Mounted Displays Using Circular Touchpad. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, IEEE, 692–703. https://doi.org/10.1109/VR46266.2020.00092
[41] Haiyan Jiang, Dongdong Weng, Zhenliang Zhang, Yihua Bao, Yufei Jia, and Mengman Nie. 2018. HiKeyb: High-Efficiency Mixed Reality System for Text Entry. In *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE, IEEE, 132–137. https://doi.org/10.1109/ISMAR-Adjunct.2018.00051
[42] Haiyan Jiang, Dongdong Weng, Zhenliang Zhang, and Feng Chen. 2019. HiFinger: One-Handed Text Entry Technique for Virtual Environments Based on Touches Between Fingers. *Sensors* 19, 14 (2019), 3063. https://doi.org/10.3390/s19143063
[43] Sunjun Kim, Jeongmin Son, Geehyuk Lee, Hwan Kim, and Woohun Lee. 2013. TapBoard: Making a Touch Screen Keyboard More Touchable. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 553–562. https://doi.org/10.1145/2470654.2470733
[44] Pascal Knierim, Valentin Schwind, Anna Maria Feit, Florian Nieuwenhuizen, and Niels Henze. 2018. Physical Keyboards in Virtual Reality: Analysis of Typing Performance and Effects of Avatar Hands. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–9. https://doi.org/10.1145/3173574.3173919
[45] Per-Ola Kristensson and Shumin Zhai. 2004. SHARK2: a Large Vocabulary Shorthand Writing System for Pen-Based Computers. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*. 43–52. https://doi.org/10.1145/1029632.1029640
[46] Falko Kuester, Michelle Chen, Mark E Phair, and Carsten Mehring. 2005. Towards Keyboard Independent Touch Typing in VR. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. 86–95. https://doi.org/10.1145/1101616.1101635
[47] Chandan Kumar, Ramin Hedeshy, Scott MacKenzie, and Steffen Staab. 2020. TAGSwipe: Touch Assisted Gaze Swipe for Text Entry. (2020). https://doi.org/abs/10.1145/3313831.3376317
[48] Andrew Kurauchi, Wenxin Feng, Ajjen Joshi, Carlos Morimoto, and Margrit Betke. 2016. EyeSwipe: Dwell-Free Text Entry Using Gaze Paths. 1952–1956. https://doi.org/10.1145/2858036.2858335
[49] Andrew Toshiaki Nakayama Kurauchi. 2018. *EyeSwipe: Text Entry Using Gaze Paths*. Ph.D. Dissertation. Universidade de São Paulo.
[50] Lik Hang Lee, Kit Yung Lam, Tong Li, Tristan Braud, Xiang Su, and Pan Hui. 2019. Quadmetric Optimized Thumb-to-Finger Interaction for Force Assisted One-Handed Text Entry on Mobile Headsets. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 3 (2019), 1–27. https://doi.org/10.1145/3351252
[51] Minkyung Lee, Woontack Woo, et al. 2003. ARKB: 3D Vision-Based Augmented Reality Keyboard. In *ICAT*. https://doi.org/10.7537/marslsj1010s13.45
[52] Seongil Lee, Sang Hyuk Hong, and Jae Wook Jeon. 2002. Designing a Universal Keyboard Using Chording Gloves. *ACM SIGCAPH Computers and the Physically Handicapped* 73-74 (2002), 142–147. https://doi.org/abs/10.1145/960201.957230
[53] Luis A Leiva, Alireza Sahami, Alejandro Catala, Niels Henze, and Albrecht Schmidt. 2015. Text entry on tiny qwerty soft keyboards. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 669–678.
[54] Frank Chun Yat Li, Richard T Guy, Koji Yatani, and Khai N Truong. 2011. The 1line Keyboard: a QWERTY Layout in a Single Line. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. 461–470. https://doi.org/10.1145/2047196.2047257
[55] Jia-Wei Lin, Ping-Hsuan Han, Jiun-Yu Lee, Yang-Sheng Chen, Ting-Wei Chang, Kuan-Wen Chen, and Yi-Ping Hung. 2017. Visualizing the Keyboard in Virtual Reality for Enhancing Immersive Experience. In *ACM SIGGRAPH 2017 Posters*. 1–2. https://doi.org/10.1145/3102163.3102175
[56] Yi Liu, Chi Zhang, Chonho Lee, Bu-Sung Lee, and Alex Qiang Chen. 2015. Gazetry: Swipe Text Typing Using Gaze. In *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction*. 192–196. https://doi.org/10.1145/2838739.2838804
[57] Xueshi Lu, Difeng Yu, Hai-Ning Liang, Xiyu Feng, and Wenge Xu. 2019. DepthText: Leveraging Head Movements Towards the Depth Dimension for Hands-Free Text Entry in Mobile Virtual Reality Systems. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, IEEE, 1060–1061. https://doi.org/10.1109/VR.2019.8797901
[58] Yiqin Lu, Chun Yu, Xin Yi, Yuanchun Shi, and Shengdong Zhao. 2017. Blindtype: Eyes-Free Text Entry on Handheld Touchpad by Leveraging Thumb's Muscle Memory. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 2 (2017), 1–24. https://doi.org/10.1145/3090083
[59] Christof Lutteroth, Moiz Penkar, and Gerald Weber. 2015. Gaze vs. Mouse: a Fast and Accurate Gaze-Only Click Alternative. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 385–394. https://doi.org/10.1145/2807442.2807461
[60] Xinyao Ma, Zhaolin Yao, Yijun Wang, Weihua Pei, and Hongda Chen. 2018. Combining Brain-Computer Interface and Eye Tracking for High-Speed Text Entry in Virtual Reality. In *23rd International Conference on Intelligent User Interfaces*. 263–267. https://doi.org/abs/10.1145/3172944.3172988
[61] I Scott MacKenzie, Hedy Kober, Derek Smith, Terry Jones, and Eugene Skepner. 2001. LetterWise: Prefix-Based Disambiguation for Mobile Text Input. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*. 111–120. https://doi.org/10.1145/502348.502365
[62] I Scott MacKenzie and R William Soukoreff. 2003. Phrase Sets for Evaluating Text Entry Techniques. In *CHI'03 Extended Abstracts on Human Factors in Computing Systems*. 754–755. https://doi.org/10.1145/765891.765971
[63] I Scott MacKenzie and Shawn X Zhang. 1999. The Design and Evaluation of a High-Performance Soft Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 25–31. https://doi.org/10.1145/302979.302983
[64] I Scott MacKenzie and Xuang Zhang. 2008. Eye Typing Using Word and Letter Prediction and a Fixation Algorithm. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications*. 55–58. https://doi.org/10.1145/1344471.1344484
[65] Päivi Majaranta, Ulla-Kaija Ahola, and Oleg Špakov. 2009. Fast Gaze Typing With an Adjustable Dwell Time. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 357–360. https://doi.org/10.1145/1518701.1518758
[66] Päivi Majaranta and Kari-Jouko Räihä. 2007. Text Entry by Gaze: Utilizing Eye-Tracking. *Text Entry Systems: Mobility, Accessibility, Universality* (2007), 175–187. https://doi.org/abs/10.1145/3313831.3376317
[67] Anders Markussen, Mikkel Rønne Jakobsen, and Kasper Hornbæk. 2014. Vulture: a Mid-Air Word-Gesture Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1073–1082. https://doi.org/10.1145/2556288.2556964
[68] Brian McElree and Marisa Carrasco. 1999. The Temporal Dynamics of Visual Search: Evidence for Parallel Processing in Feature and Conjunction Searches. *Journal of Experimental Psychology: Human Perception and Performance* 25, 6 (1999), 1517. https://doi.org/10.1037/0096-1523.25.6.1517
[69] Mark McGill, Daniel Boland, Roderick Murray-Smith, and Stephen Brewster. 2015. A Dose of Reality: Overcoming Usability Challenges in VR Head-Mounted Displays. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2143–2152. https://doi.org/10.1145/2702123.2702382
[70] Martez E Mott, Shane Williams, Jacob O Wobbrock, and Meredith Ringel Morris. 2017. Improving Dwell-Based Gaze Typing With Dynamic, Cascading Dwell Times. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2558–2570. https://doi.org/10.1145/3025453.3025517
[71] Peter Norvig. 2013. English Letter Frequency Counts: Mayzner Revisited or ETAOIN SRHLDCU. (2013).
[72] Midori Ohkita, Yoshie Obayashi, and Masako Jitsumori. 2014. Efficient Visual Search for Multiple Targets Among Categorical Distractors: Effects of Distractor-Distractor Similarity Across Trials. *Vision Research* 96 (2014), 96–105. https://doi.org/10.1016/j.visres.2014.01.009
[73] Jakob Olofsson. 2017. Input and Display of Text for Virtual Reality Head-Mounted Displays and Hand-Held Positionally Tracked Controllers.
[74] Alexander Otte, Tim Menzner, Travis Gesslein, Philipp Gagel, Daniel Schneider, and Jens Grubert. 2019. Towards Utilizing Touch-Sensitive Physical Keyboards for Text Entry in Virtual Reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, IEEE, 1729–1732. https://doi.org/10.1109/VR.2019.8797740
[75] Antti Oulasvirta, Anna Reichel, Wenbin Li, Yan Zhang, Myroslav Bachynskyi, Keith Vertanen, and Per Ola Kristensson. 2013. Improving Two-Thumb Text Entry on Touchscreen Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2765–2774. https://doi.org/10.1145/2470654.2481383

[76] Farshid Salemi Parizi, Eric Whitmire, and Shwetak Patel. 2019. AuraRing: Precise Electromagnetic Finger Tracking. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 4 (2019), 1–28. https://doi.org/10.1145/3369831

[77] Diogo Pedrosa, Maria Da Graça Pimentel, Amy Wright, and Khai N Truong. 2015. Filteryedping: Design Challenges and User Performance of Dwell-Free Eye Typing. *ACM Transactions on Accessible Computing (TACCESS)* 6, 1 (2015), 1–37. https://doi.org/10.1145/2724728

[78] Abdul Moiz Penkar, Christof Lutteroth, and Gerald Weber. 2012. Designing for the Eye: Design Parameters for Dwell in Gaze Interaction. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*. 479–488. https://doi.org/10.1145/2414536.2414609

[79] Ken Perlin. 1998. Quikwriting: Continuous Stylus-Based Text Entry. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*. 215–216. https://doi.org/10.1145/288392.288613

[80] Duc-Minh Pham and Wolfgang Stuerzlinger. 2019. HawKEY: Efficient and Versatile Text Entry for Virtual Reality. In *25th ACM Symposium on Virtual Reality Software and Technology*. 1–11. https://doi.org/10.1145/3359996.3364265

[81] Ryan Qin, Suwen Zhu, Yu-Hao Lin, Yu-Jung Ko, and Xiaojun Bi. 2018. Optimal-T9: an Optimized T9-Like Keyboard for Small Touchscreen Devices. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*. 137–146. https://doi.org/10.1145/3279778.3279786

[82] Philip Quinn and Shumin Zhai. 2016. A cost-benefit study of text entry suggestion interaction. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. 83–88.

[83] Vijay Rajanna and John Paulin Hansen. 2018. Gaze Typing in Virtual Reality: Impact of Keyboard Design, Selection Method, and Motion. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. 1–10. https://doi.org/10.1145/3204493.3204541

[84] Mark Richardson, Matt Durasoff, and Robert Wang. 2020. Decoding Surface Touch Typing From Hand-Tracking. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 686–696. https://doi.org/10.1145/3379337.3415816

[85] Jochen Rick. 2010. Performance Optimizations of Virtual Keyboards for Stroke-Based Text Entry on a Touch-Based Tabletop. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*. 77–86. https://doi.org/10.1145/1866029.1866043

[86] Sherry Ruan, Jacob O Wobbrock, Kenny Liou, Andrew Ng, and James A Landay. 2018. Comparing Speech and Keyboard Text Entry for Short Messages in Two Languages on Touchscreen Phones. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (2018), 1–23. https://doi.org/10.1145/3161187

[87] Sayan Sarcar, Prateek Panwar, and Tuhin Chakraborty. 2013. EyeK: an Efficient Dwell-Free Eye Gaze-Based Text Entry System. In *Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction*. 215–220. https://doi.org/pdf/10.1145/2525194.2525288

[88] Weinan Shi, Chun Yu, Xin Yi, Zhen Li, and Yuanchun Shi. 2018. TOAST: Ten-Finger Eyes-Free Typing on Touchable Surfaces. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 1–23. https://doi.org/10.1145/3191765

[89] Shyamli Sindhwani, Christof Lutteroth, and Gerald Weber. 2019. ReType: Quick Text Editing With Keyboard and Gaze. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13. https://doi.org/10.1145/3290605.3300433

[90] Brian A Smith, Xiaojun Bi, and Shumin Zhai. 2015. Optimizing Touchscreen Keyboards for Gesture Typing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 3365–3374. https://doi.org/10.1145/2702123.2702357

[91] R William Soukoreff and I Scott MacKenzie. 2003. Metrics for Text Entry Research: an Evaluation of MSD and KSPC, and a New Unified Error Metric. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 113–120. https://doi.org/10.1145/642611.642632

[92] Marco Speicher, Anna Maria Feit, Pascal Ziegler, and Antonio Krüger. 2018. Selection-Based Text Entry in Virtual Reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13. https://doi.org/10.1145/3173574.3174221

[93] Srinath Sridhar, Anna Maria Feit, Christian Theobalt, and Antti Oulasvirta. 2015. Investigating the Dexterity of Multi-Finger Input for Mid-Air Text Entry. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 3643–3652. https://doi.org/10.1145/2702123.2702136

[94] Bruce H Thomas and Wayne Piekarski. 2002. Glove Based User Interaction Techniques for Augmented Reality in an Outdoor Environment. *Virtual Reality* 6, 3 (2002), 167–180. https://doi.org/10.1145/988834.988871

[95] Keith Vertanen, Crystal Fletcher, Dylan Gaines, Jacob Gould, and Per Ola Kristensson. 2018. The Impact of Word, Multiple Word, and Sentence Input on Virtual Keyboard Decoding Performance. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12. https://doi.org/10.1145/3173574.3174200

[96] Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. VelociTap: Investigating Fast Mobile Text Entry Using Sentence-Based Decoding of Touchscreen Keyboard Input. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 659–668. https://doi.org/10.1145/2702123.2702135

[97] James Walker, Bochao Li, Keith Vertanen, and Scott Kuhl. 2017. Efficient Typing on a Visually Occluded Physical Keyboard. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 5457–5461. https://doi.org/10.1145/3025453.3025783

[98] Junjue Wang, Kaichen Zhao, Xinyu Zhang, and Chunyi Peng. 2014. Ubiquitous Keyboard for Small Mobile Devices: Harnessing Multipath Fading for Fine-Grained Keystroke Localization. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*. 14–27. https://doi.org/10.1145/2594368.2594384

[99] David J Ward, Alan F Blackwell, and David JC MacKay. 2000. Dasher—a Data Entry Interface Using Continuous Gestures and Language Models. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*. 129–137. https://doi.org/10.1145/354401.354427

[100] Pierre Weill-Tessier, Jayson Turner, and Hans Gellersen. 2016. How do you look at what you touch? A study of touch interaction and gaze correlation on tablets. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*. 329–330.

[101] Eric Whitmire, Mohit Jain, Divye Jain, Greg Nelson, Ravi Karkar, Shwetak Patel, and Mayank Goel. 2017. Digitouch: Reconfigurable Thumb-to-Finger Input and Text Entry on Head-Mounted Displays. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 1–21. https://doi.org/10.1145/3130978

[102] Pui Chung Wong, Kening Zhu, and Hongbo Fu. 2018. Fingert9: Leveraging Thumb-to-Finger Interaction for Same-Side-Hand Text Entry on Smartwatches. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–10. https://doi.org/10.1145/3173574.3173752

[103] Wenge Xu, Hai-Ning Liang, Anqi He, and Zifan Wang. 2019. Pointing and Selection Methods for Text Entry in Augmented Reality Head Mounted Displays. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, IEEE, 279–288. https://doi.org/10.1109/ISMAR.2019.00026

[104] Wenge Xu, Hai-Ning Liang, Yuxuan Zhao, Tianyu Zhang, Difeng Yu, and Diego Monteiro. 2019. RingText: Dwell-Free and Hands-Free Text Entry for Mobile Head-Mounted Displays Using Head Motions. *IEEE Transactions on Visualization and Computer Graphics* 25, 5 (2019), 1991–2001. https://doi.org/10.1109/TVCG.2019.2898736

[105] Zheer Xu, Weihao Chen, Dongyang Zhao, Jiehui Luo, Te-Yen Wu, Jun Gong, Sicheng Yin, Jialun Zhai, and Xing-Dong Yang. 2020. BiTipText: Bimanual Eyes-Free Text Entry on a Fingertip Keyboard. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13. https://doi.org/fullHtml/10.1145/3313831.3376306

[106] Naoki Yanagihara, Buntarou Shizuki, and Shin Takahashi. 2019. Text Entry Method for Immersive Virtual Environments Using Curved Keyboard. In *25th ACM Symposium on Virtual Reality Software and Technology*. 1–2. https://doi.org/abs/10.1145/3173574.3174221

[107] Xin Yi, Chen Wang, Xiaojun Bi, and Yuanchun Shi. 2020. PalmBoard: Leveraging Implicit Touch Pressure in Statistical Decoding for Indirect Text Entry. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13. https://doi.org/abs/10.1145/3313831.3376441

[108] Xin Yi, Chun Yu, Mingrui Zhang, Sida Gao, Ke Sun, and Yuanchun Shi. 2015. Atk: Enabling Ten-Finger Freehand Typing in Air Based on 3d Hand Tracking Data. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 539–548. https://doi.org/pdf/10.1145/2807442.2807504

[109] Yafeng Yin, Qun Li, Lei Xie, Shanhe Yi, Edmund Novak, and Sanglu Lu. 2016. CamK: a Camera-Based Keyboard for Small Mobile Devices. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, IEEE, 1–9. https://doi.org/abs/10.1109/INFOCOM.2016.7524400

[110] Chun Yu, Yizheng Gu, Zhican Yang, Xin Yi, Hengliang Luo, and Yuanchun Shi. 2017. Tap, Dwell or Gesture? Exploring Head-Based Text Entry Techniques for HMDs. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 4479–4488. https://doi.org/10.1145/3025453.3025964

[111] Chun Yu, Ke Sun, Mingyuan Zhong, Xincheng Li, Peijun Zhao, and Yuanchun Shi. 2016. One-Dimensional Handwriting: Inputting Letters and Words on Smart Glasses. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 71–82. https://doi.org/10.1145/2858036.2858542

[112] Difeng Yu, Kaixuan Fan, Heng Zhang, Diego Monteiro, Wenge Xu, and Hai-Ning Liang. 2018. PizzaText: Text Entry for Virtual Reality Systems Using Dual Thumbsticks. *IEEE Transactions on Visualization and Computer Graphics* 24, 11 (2018), 2927–2935. https://doi.org/10.1109/TVCG.2018.2868581

[113] Shumin Zhai, Michael Hunter, and Barton A Smith. 2002. Performance Optimization of Virtual Keyboards. *Human-Computer Interaction* 17, 2-3 (2002), 229–269. https://doi.org/10.1145/1866029.1866043

[114] Mingrui Ray Zhang, He Wen, and Jacob O Wobbrock. 2019. Type, Then Correct: Intelligent Text Correction Techniques for Mobile Text Entry Using Neural

Networks. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 843–855. https://doi.org/10.1145/3332165.3347924

[115] Suwen Zhu, Tianyao Luo, Xiaojun Bi, and Shumin Zhai. 2018. Typing on an Invisible Keyboard. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13. https://doi.org/10.1145/3173574.3174013